AUSTRALIAN OS9 NEWSLETTER

EDITOR	Gordon Bentzen	(07) 344-3881
SUB-EDITOR	Bob Devries	(07) 372-7816
TREASURER	Don Berrie	(07) 375-1284
LIBRARIAN	Jean-Pierre Jacquet	(07) 372-4675
SUPPORT	Brisbane OS9 Users Group	

Addresses for Correspondence

Editorial Material:

Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

Subscriptions & Library Requests:

Jean-Pierre Jacquet
27 Hampton Street
DURACK Qld 4077

Volume 5 June 1991 Number 5

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 5 Number 5

EDITOR : Gordon Bentzen
SUBEDITOR : Bob Devries
TREASURER : Don Berrie
LIBRARIAN : Jean-Pierre Jacquet
SUPPORT : Brisbane OS9 Level 2 Users Group.

Here is the latest! The U.S. OS9 Usergroup has folded. We can again thank Tandy for dropping the CoCo and Microware for dropping 6809 OS9.
The brilliant marketing decision by Tandy to discontinue the CoCo has certainly left a large number of users, worldwide, out in the cold. Their strategy was that CoCo users would "upgrade" to their Messy DOS machines. Whilst their MS-DOS based machines are no doubt of good quality, I am sure that very few ex CoCo'ers rushed to buy a Tandy MS-DOS machine, certainly not OS9'ers anyway.
The demise of the U.S. Usergroup leaves the European OS9 Usergroup and our own National OS9 Usergroup as the only active OS9 groups that we are aware of. From messages on the CoCo Listserver in the States, it seems that there is at least interest in the Australian Usergroup. So we may get some new members from the U.S.A.
Despite the lack of support from both Tandy and Microware, we believe that there are still a couple of years' life in the CoCo and OS9 Level 2. The obvious problem will be the availability of the CoCo OS9 operating system and other software utilities and applications. Until now, applications specifically for CoCo OS9 have continued to grow through third party developers, but this will slow to a stop as time goes on.
I do apologize to those "Natgroup" members running OSK, for our continued concentration on Tandy's CoCo, but it is a fact that CoCo users are at the moment in the majority.
So where do we go from here? Firstly, to the new subscribers and those just beginning the OS9 experience, **don't give up!** because we are not going to. We do have some new members who are just beginning to tackle OS9, and we encourage you to continue. The knowledge gained will serve you well in the future, whatever that may be.
MS-DOS is not the alternative for any OS9'er. OSK is the obvious way to go! The question is, what platform do we use, and what will be the cost?
We have a few members running OSK on ATARI machines, which may be an alternative, but Atari have not done much with hardware upgrades lately, and the OSK operating system for it is expensive. I believe that there is a "windows" environment also available if you have plenty of money. Perhaps someone out there could set me straight on this if I have it wrong.
There is also an OSK port for the AMIGA but for most hobbyists the price of this is just out of the question.
It seemed to us that the much talked about MM/1 would be the way to go, but it is starting to look like just a lot of talk still. Just try to buy one and see how you go. From the specs we have seen the MM/1, complete with the OSK operating system, looks a good package, at good value. It offers Multimedia computing, high resolution graphics, stereo sound, plus the multi-tasking and window environment with which we are familiar. If "Interactive Media Systems" are going to get this machine off the ground, they certainly need to improve their product availability and distribution.
What about OS/9000?? The price of P.C.'s has fallen in recent times, so much so, that a 80386 SX machine is now affordable, especially if you shop around. Add to the hardware price about $1000.00 (I believe) for the OS/9000 operating system and you can be up and running in a familiar OS9 environment. So this option may well be worth some consideration. What do you think??
Well, for the time being at least, I have just convinced myself that my CoCo3's with OS9 Level 2 are in for a workout for a while yet.
<u>OS9 Level 2</u> At a newsletter meeting last week, we were contemplating the running of stock standard OS9 level 2, as it was distributed by Tandy, and the thought was rather alarming. The Level 2 system we are now running has so many "patched" system modules that it hardly represents the original. So in the near future we will document all those patches which have improved the performance in both speed and reliability.

Until next time; Cheers, Gordon.

A Basic09 Tutorial
by Bob Devries

Another problem area for programmers recently converted to Basic09 is the TYPE statement. This is used when a programmer needs to lump together several variables to be referred to as one unit. Let me give you an example. Say I want to write a little database programme (as my earlier version in C) to keep names, addresses, and phone numbers. Here's what the start of the programme would look like:

```
PROCEDURE Program
TYPE            record=surname:STRING[20];
firstname:STRING[20]; street:STRING [20];
city:STRING[20];          state:STRING[3];
postcode:INTEGER;     area:STRING    [3];
phone:STRING[7]
DIM address:record
```

So here I have the same database record as I have used previously (a series of articles starting March 1989). A database entry is a complex variable called address of TYPE record. That is, the variable record has in it all the variables referred to in record. So to refer to the city field in the database entry, I would call it address.city, easy see ?

If I want to fill each of the variables of the complex variable address, I could do this:

```
address.surname = "DEVRIES"
address.firstname = "BOB"
address.street = "21 Virgo Street"
address.city = "INALA"
address.state = "Qld"
address.postcode = 4077
address.area = "07"
address.phone = "3727816"
```

If I want to write a database entry to a disk file, I would merely do this:

```
PUT #file, address
```

This will put all the variables which make up the complex variable address into the diskfile one after the other. Of course the diskfile must have been opened first.

Similarly, to read an existing entry from a diskfile, I would use this line:

```
GET #file, address
```

If the disk file was 20 entries long, and I wanted to get the fifteenth one, I would first seek to the fourteenth (all records start at the zeroeth) record like this:

```
SEEK #file,14 * SIZE(address)
```

Then I would read the entry as before.

Here is a sample piece of programme which sets up the database record in memory using TYPE, and fills it, and then displays it in an overlay window. One thing you should be aware of, you MUST initialise variables in Basic09, because all variables are filled with garbage after being dimensioned.

```
PROCEDURE Program
TYPE            record=surname:STRING[20];
firstname:STRING[20]; street:STRING [20];
city:STRING[20];          state:STRING[3];
postcode:INTEGER;     area:STRING    [3];
phone:STRING[7]
DIM address:record
DIM file:INTEGER
DIM a:STRING[1]

PRINT CHR$(12)

address.surname="Bentzen"
address.firstname="Gordon"
address.street="8 Odin Street"
address.city="Sunnybank"
address.state="Qld"
address.postcode=4109
address.area="07"
address.phone="3443881"

RUN gfx2("OWSet",1,9,4,32,11,1,0)
RUN gfx2("OWSet",0,10,5,30,9,0,1)

PRINT "Surname:";
PRINT address.surname
PRINT "Firstname:";
PRINT address.firstname
PRINT "Street:";
PRINT address.street
PRINT "City:";
PRINT address.city
PRINT "State:";
PRINT address.state
PRINT "Postcode:";
PRINT USING "i5",address.postcode
PRINT "Area code:";
```

```
PRINT address.area
PRINT "Phone:";
PRINT address.phone
a=""
WHILE a="" DO
RUN inkey(a)
ENDWHILE

OPEN #file,"DATABASE":UPDATE
SEEK #file,0
PUT #file,address
CLOSE #file

RUN gfx2("OWEnd")
RUN gfx2("OWEnd")
```

I'll give you a run-down on what is in this programme.

First, the TYPE command, setting up the memory image of the database record. Next, dimension the complex variable, as well as some other useful variables. Then I filled the various parts of the complex variable with data for one record of the database with Gordon's name, address etc. No doubt you'll understand now that to access each part of the database record, its identifier is 'address.xxxxxx' where the x's are the various sections of the record, e.g. city.

Now to display the record, I open an overlay window big enough to display the fields of the record, and print them. Notice the use of PRINT USING for the postcode field. For this, you must use a format length of one more than the length of the variable, hence, 'i5'. Next I wait for a keypress before closing the overlay, and writing the record, and quitting. In this example, the record is always written to position zero of the file.

Next month, I'll show you how to convert programmes from other BASIC languages, including RSBasic, GWBasic etc. I'll include a working example, in both the original format, and the converted Basic09 programme.

Regards,
Bob Devries

ooooooooooOOOOOOOOOOoooooooooo

C Tutorial
Chapter 1 - Getting Started

WHAT IS AN IDENTIFIER?

Before you can do anything in any language, you must at least know how you name an identifier. An identifier is used for any variable, function, data definition, etc. In the programming language C, an identifier is a combination of alphanumeric characters, the first being a letter of the alphabet or an underline, and the remaining being any letter of the alphabet, any numeric digit, or the underline. Two rules must be kept in mind when naming identifiers.

1. The case of alphabetic characters is significant. Using "INDEX" for a variable is not the same as using "index" and neither of them is the same as using "InDeX" for a variable. All three refer to different variables.
2. As C is defined, up to eight significant characters can be used and will be considered significant. If more than eight are used, they may be ignored by the compiler. This may or may not be true of your compiler. You should check your reference manual to find out how many characters are significant for your compiler.

It should be pointed out that some C compilers allow use of a dollar sign in an identifier name, but since it is not universal, it will not be used anywhere in this tutorial. Check your documentation to see if it is permissible for your particular compiler.

WHAT ABOUT THE UNDERLINE?

Even though the underline can be used as part of a variable name, it seems to be used very little by experienced C programmers. It adds greatly to the readability of a program to use descriptive names for variables and it would be to your advantage to do so. Pascal programmers tend to use long descriptive names, but most C programmers tend to use short cryptic names. Most of the example programs in this tutorial use very short names for that reason.

Any computer program has two entities to consider, the data and the program. They are

highly dependent on one another and careful planning of both will lead to a well planned and well written program. Unfortunately, it is not possible to study either completely without a good working knowledge of the other. For this reason, this tutorial will jump back and forth between teaching methods of program writing and methods of data definition. Simply follow along and you will have a good understanding of both. Keep in mind that, even though it seems expedient to sometimes jump right into the program coding, time spent planning the data structures will be well spent and the final program will reflect the original planning.

HOW THIS TUTORIAL IS WRITTEN

As you go through the example programs, you will find that every program is complete. There are no program fragments that could be confusing. This allows you to see every requirement that is needed to use any of the features of C as they are presented. Some tutorials I have seen give very few, and very complex examples. They really serve more to confuse the student. This tutorial is the complete opposite because it strives to cover each new aspect of programming in as simple a context as possible. This method, however, leads to a lack of knowledge in how the various parts are combined. For that reason, the last chapter is devoted entirely to using the features taught in the earlier chapters. It will illustrate how to put the various features together to create a usable program. They are given for your study, and are not completely explained. Enough details of their operation are given to allow you to understand how they work after you have completed all of the previous lessons.

A DISCUSSION OF SOME OF THE FILES

CCL.BAT

[ED: this part refers to files on the original MSDos distribution, and can only be used on computers which run MSDos. These files are, however, included in the c_tutor.ar file on PD disk number !!!]

This file, which does not exist on the distribution disk, is the batch file that calls in an editor, then the compiler (Pass 1 and Pass 2, if it exists), and finally runs the resulting compiled program. There are several examples of batch files which can be used with various compilers given in the "COMPILER.DOC" file on the distribution diskette. It is up to you to type in a batch file for use with your particular compiler, considering also the method required to call in your editor. To use it, simply type the batchfile name with the desired filename. After typing in your particular CCL.BAT file, try it by typing CCL FIRSTEX. You will get the source file displayed on the monitor by your editor. If you don't have one of the compilers listed in the "COMPILER.DOC" file, you will have to modify the batch file for your particular compiler.

The pass or passes of the compiler will be executed, followed by the linking process. The final program will be loaded and run, then the files generated by the process will be erased to prevent filling the disk up.
If you have a hard disk available, it will be up to you to modify the batch file to perform the above described operations. Even though you will have a lot of files to compile and run, you will find that a batch file similar to this will do most of the work for you and you will proceed very quickly.
In order to do the programming exercises, you will need to go through the same steps as when running the example programs. This is simple to do by simply typing your own filename with the CCL program call. It is highly recommended that you do the programming exercises to gain the programming experience.

LIST.EXE

This file will list the source files for you with line numbers and filename. To use it, simply type "LIST" followed by the appropriate filename. Type LIST FIRSTEX.C now for an example. The C source code is given later in Chapter 14 along with a brief description of its operation.

PRINTALL.BAT

This is a batch file that will call the above LIST.EXE file once for each of the example C programs, printing all of the files out. If you want a hardcopy of all of the files, enter PRINTALL and watch as your printer fills about 150 sheets of paper with C programs.

oooooooooOOOOOOOOOoooooooooo

ERRATA ERRATA ERRATA ERRATA ERRATA ERRATA

An error exists in the listing for the programme 'DIRL' in last month's newsletter. On page 9, the sixth last line of code, currently reads:

```
case S_BYTES:   dsize=dirsize+padding;      break
```

To make this compile correctly, you must add a semi-colon (;) to the end of the line (after 'break').

Sorry about that, we didn't catch that one in time. ED.

ERRATA ERRATA ERRATA ERRATA ERRATA ERRATA

Problems with Dirl.c

A number of our readers have had some problems with the utility dirl, that we offered in the last newsletter. The following information will be of value to those of you who may have been having trouble.

1. In the line:
"case S_BYTES: dsize=dirsize+padding; break"
on page 9 of the newsletter, there is a missing ";" after the word break.

2. The lseek call, also on page 9 ... is NOT
lseek(path,64l,0)

it should be typed lseek(path,64L,0);
/\
Note that this character is an alpha 'ell', either upper or lower case, to signify a long. See page 3-22 of the Microware C Compiler manual for details.

3. If you happen to compile this programme, and automatically include the cgfx.l library in your linker list, you will find that the programme compiles OK, but will not run. See below for details.

The problems arises because a function is defined in more than one library. When this happens, the linker does not report an error, as there really is no error condition generated, but problems can arise simply because of the logic within the code. The function that is used for linking in these cases, is the first function of that name found. For example in Mike Sweet's CGFX7 library, he defines a function, read(). This function is also defined in the standard clib.l library, and as well, in the Kreider clib.l library.

The trouble stems from the fact that the functions return different values on reaching EOF. (The kreider library and the standard library read() functions return 0 on end of file, but the function from the cgfx7 library returns -1). If you read the manual, you'll find that, in their own way, they both are correct! If you tried to compile the dirl.c programme that was printed in the last newsletter (and got past the typographical error mentioned above!), and included cgfx.l (Mike Sweet's version), in the link list, the programme would compile OK, but it will not run correctly.

The solution is to simply exclude the cgfx.l library when performing the final link step, and all will be well.

Also note that you need to use the Kreider clib.l library, rather than the standard clib.l library, as some of the functions called do not exist in the standard clib.l.

CoCo-Link

CoCo-Link is an excellent magazine to help you with the RS00S side of the Colour Computer. It is a bi-monthly magazine published by Mr. Robbie Dalzell. Send your subscriptions to:

CoCo-Link
31 Nedlands Crescent
Pt. Noarlunga Sth.
South Australia
Phone: (08) 3861647

Clusters

by Brian White

[ED: The following article was posted to the INTERNET CoCo OS9 discussion list by Brian White in the USA. We think that readers who do not have access to that list may find it an interesting article, particularly those with hard drives. The article is of a fairly technical nature, and if you have any problems in understanding what is being talked about, then maybe it would be best if you did not attempt to make any of the changes about which he writes. <Grin>.

We would like to take this opportunity to thank Brian for the use of this article.]

Clusters

Well, I did some poking around into "clusters", what they mean, and how to use them, so I decided to post it here for you and anyone else who is interested.

First, a sector is always a sector, and an LSN (Logical Sector Number) is always an LSN. These two terms should never be confused with the word "cluster". The RBF manager that comes with OS-9 can access up to 2^24 or 16,777,216 unique 256-byte sectors for a grand total of 2^32 or 4,294,967,296 bytes (4 GigaBytes)... before needing to undergo partitioning :-) Therefore, LSN-0 is always 256 bytes, a file-descriptor is always 256 bytes, etc., etc. Anywhere these terms are used in the OS-9 manuals, they are used correctly.

A "cluster" is a lot like the minimum sector allocation size except that it is the same for all types of files and it can't be changed on the fly. If the cluster size is set to 32, every file will allocate a multiple of 32 sectors (the first segment is actually 31 sectors plus 1 file-descriptor sector) and remain there until it needs another cluster.

As far as how cluster size and sector allocation size (SAS) work together, the initial file size is always 1 cluster. On disks with 1-sector clusters, this is used exclusively by the file-descriptor. After that, as soon as data is written past the end of a segment, more space is allocated to the file in chunks of SAS rounded up to the next cluster size. (eg. if cluster size=$08 and SAS=$23, additional space is allocated in chunks of $28 sectors). When the file is closed, it is shrunk to the least used number of clusters.

Changing the cluster size is not as easy as you might think. There is nothing about it in any of my device descriptors and 'format' has no option to that effect. To change it manually, you must:

- FORMAT the disk normally
- dEd/QTip the disk:
- edit LSN-0:
- change the cluster size to any power of 2 (bytes $06-$07)
- divide the number of bytes in the allocation bitmap (bytes $04-$05) by the same power of 2 used above and round down to the next nearest byte.
- edit root directory file-desciptor: (LSN of root fd is in LSN-0)
- change the size of the first and only segment so the directory ends at the end of a cluster. For example, my hard disk has its root directory file-descriptor on sector $50 and the root directory itself starts on sector $51. If I wanted a cluster size of 32 ($20), the start of the next cluster would be $60, so I change the size of the root directory's segment, originally set to $07 by 'format', to $0F ($60-$51=$0F). If a cluster is not is not completely allocated within the file structure, dCheck assumes that none of it is allocated. OS-9/RBF will take care of all this allocation for you once the required manual setup is finished.
- perform a dCheck on the disk
- use a utility like BD/BA from the RePack software package to deallocate any clusters that dCheck mentions (or fool around with the first few bytes of LSN-1 by hand using dEd/QTip until dCheck reports a clean disk). DCheck always reports clusters by the LSN they start on. BA and BD handle these LSN's

properly to fiddle with the correct bits in the allocation map.

- You now have a disk with whatever cluster size you put in bytes $06-$07.

(This is really easier than it sounds... honest!)

While playing around with this, I discovered that 'dCheck' seems to have a small bug in it that causes it not to check the last byte of the allocation bitmap! So if you start playing around with more than is said above and find that dCheck is not bothering to mention the $400 allocated sectors at the end of the disk that should be free, it's dCheck's fault, not yours. RBF seems to handle everything just fine. 'Free' has a similar bug which causes it to analyze the entire last byte even if some of it's bits aren't used. Rounding down LSN0's bytes $04-$05 will cause up to 7 clusters never to be used, but will work around both of these bugs.

I have moved files to my disk after setting it to various cluster sizes and have had no problems with OS-9/RBF in any way. This wonderful feature seems to be fully implemented except for turning it on.

Anyone who edits over multiple windows, saving from all of them often, knows that the files fragment into many 1-sector segments quite quickly. This cluster size can help solve that problem because a 1-sector file in a 32-sector cluster still has lots of space to expand into before another segment is needed.

With the upcoming wide-spread use of OS-9/68000 (or OS-k), RBF has an added feature of being able to handle devices with sector sizes of larger than 256 bytes. How this will be handled, I don't know. I would guess, since the OS-k file structure is identical to that of OS-9, that it just internally splits each sector up into 256-byte chunks and then sets the cluster size accordingly to avoid file fragmentation within a sector. (eg a 512-byte sector would get a min cluster size of 2 while a 2048-byte sector would get a min cluster size of 8, etc.).

Hope all this is of use to someone...

Brian White.

oooooooooo0000000000oooooooooo

ModPatch Problems

Taken from the BitNet message system.

Q: Now that I can use the SCII/4inl, I tried to run the aciapak patch included with KBCOM. It claims that none of the data matches when it goes to patch. I checked the crc value in the header of the patch against the ident of the aciapak module and it seems to be the correct one. Am I missing something.

A: I think I know what the problem is. modpatch has a small bug which may be biting you. Edit the modpatch file and make sure that there is always at LEAST one character after each '*'. ie, change the lines containing only "*" so that they contain "* " and see if that helps.

As I remember, the modpatch file contains a bunch of comments, ending like:

```
*
I aciapak
c .....
```

But since there is no character following the '*', the newline is getting swallowed and "I aciapak" becomes part of the comment! And you haven't linked to a module, so of course none of the bytes match. Greg Law discovered this doozy.

oooooooooo0000000000oooooooooo

FREE - A bug-free version

written by Mark Griffith et al

[ED: This source code sample has come to us via the BitNet message system, from Mark Griffith. Because of its length, this part one. Don't miss the next issue!]

In response to:

Q: Free returned the same thing... wasn't there a bug in one version of free that returned the value for large drives wrong?

Tim K. says:

A: The version of "free" distributed with some of the GShell patches had such a bug.

That version of Free did have a buglet with large drives. I only had a 20 meg at the time to test it with. The bug was soon discovered and fixed. The fixed version has been on CIS for a few years so it should be making the BBS rounds by now (grin). I've included it here for those who don't have it.

```
*
* A new FREE command for use with Multi-Vue
*
* Gives the number of bytes free instead of
* sectors -- works just like the original FREE
* command.
*
* Syntax: Free [dev]
*
*     or  clicking on the FREE selection after popping
*         down the DISK menu from Multi-Vue will give
*         the free space remaining on the device currently
*         being used.
*
* This utility has been updated to insure an accurate count
* of the remaining disk space on devices that can store up
* to 4.3 GIGAbytes -- so it should work on any hard disk.
*
* This source is for assembling with ASM and not RMA.
* Much of this code was taken from the PD program LSH
* available elsewhere in this Forum.
*
* Mark Griffith [76070,41]
*
* Additional help from Kent Meyers (thanks Kent!)
* Minor hacks made by Bruce Isted
* Second minor hack for Total Sectors by Don Berrie 910530

        nam   FRee
        ttl   New FREE command for Multi-Vue

        ifpl
        use   /dd/DEFS/os9defs
        endc

***** REVISION and EDITION *****
```

```
edition  set   3
rev      set   reent+edition
type     set   prgrm+objct

*** The MODULE header macro ***

         mod   endmod,name,type,rev,start,endata

********    Data Area   ********

         org   0

descpt   rmb   256        Storage for path descpt
temp1    rmb   2          Scratch memory
temp2    rmb   2          Scratch memory
overflow rmb   2          Scratch memory
inpath   rmb   1          Input path number
devnam   rmb   6          Device name
total    rmb   4          Nbr sectors on disk
nfree    rmb   4          Nbr of free sectors
volname  rmb   32         Disk volume name
         rmb   200        minimum stack recommended

endata   equ   .

name     fcs   /FRee/
         fcb   edition
         pag
*************************************************************
*                                                           *
*                MAIN PROGRAM CODE                          *
*                                                           *
*************************************************************

start    equ   *

         leay  devnam,u
         lda   ,x+         Get the first character
         cmpa  #$0d        Is it a carriage return?
         beq   open        Yes, then go open the disk
         cmpa  #'/         Is it a slash?
         beq   copy        Then copy the device name

exit     os9   f$exit      Finish and exit

copy     sta   ,y+         Store the character already gotten
copy1    lda   ,x+         Get the next one
         cmpa  #'/         Is it another slash?
         beq   open
         cmpa  #$0d        Is it a carriage return
         beq   open
         cmpa  #$20        or a space?
         beq   open
         sta   ,y+         If none of the above, store it
         bra   copy1       Go get some more

open     ldd   #$400d      Get a "@CR"
```

```
        std   ,y        Append it to devname
        lda   #1        Open for read only
        leax  devnam,u  Point to the name
        os9   i$open    Open the disk as a file
        bcs   exit      Exit if an error on open
        sta   inpath,u  Store the path number

* Read in the volume name and path descriptor *

getvol  ldx   #0        Get ready to seek to LSN0
        pshs  u         save U
        tfr   x,u
        os9   i$seek    Set file pointer to LSN0
        puls  u         Restore U
        bcs   exit      Exit if an error occured
        leax  descpt,u  Point to the descp. buffer
        ldy   #256
        os9   i$read    Now read in the descriptor
        bcs   exit      Exit if an error occured

* Save total size & volume name *

        pshs  x
        leax  0,x   Point to total sectors
        leay  total,u
        ldb   #3        Set counter
        lda   #0        Dummy value 'cause 4 bytes
        sta   ,y+
svtotl0 lda   ,x+       Save char
        sta   ,y+       in buffer
        decb
        bne   svtotl0   More if not done
        puls  x         Reset pointer
        pshs  x
        leax  $1f,x     Point to volume name buffer
        leay  volname,u
        ldb   #32       Set the counter
svnaml0 lda   ,x+       Now save a character
        sta   ,y+       in the buffer
        decb
        bne   svnaml0   Get some more if not done
        puls  x

* Read free sector bit map from sector 1 *

volll   ldd   $04,x     Get nbr of bytes in map
        std   temp1,u   Store it
        ldd   $06,x     Get sectors per cluster
        std   temp2,u   and store that too
        clr   ,-s
        leax  descpt,u  Read in first part of bit map
        ldy   #256
        lda   inpath,u
        os9   i$read
        lbcs  exit      Exit if an error occured

* Add up all the 0's in the bit map *
```

```
        ldd   #0

* BRI:  minor fix, as in Level 2 U=0 always
        std   overflow,u Set overflow to zero
*       std   overflow   Set overflow to zero

vol25   ldy   #8         Nbr of bits per byte
vol30   lsr   ,x
        bcs   vol35      check bit
        addd  #1         add one
        bne   vol35
        inc   overflow+1,u
vol35   leay  -1,y       bit counter
        bne   vol30
        leax  1,x        map pointer
        ldy   templ,u
        beq   vol37
        dec   ,s
        bne   vol36
        pshs  d,y
        lda   inpath,u
        leax  descpt,u
        ldy   #256
        os9   i$read
        lbcs  exit
        puls  d,y
vol36   leay  -1,y       byte counter
        sty   templ,u
        bne   vol25
        ldy   overflow,u

* BRI:  Y is not always 0

* times sectors per cluster = 24 bits
*   (Y=0)
*

* BRI:  fixed set up for free sectors calculation, added DD.BIT check
vol37   pshs  d          save free clusters LSBs for free sectors calculation
        ldx   temp2,u    DD.BIT sectors per cluster
        beq   vol45      DD.BIT=0 illegal, go report as one sector clusters
*vol37  ldx   temp2,u    DD.BIT sectors per cluster
```

oooooooooo0000000000oooooooooo